Symbolic Algebra in Mathematical Analysis of Chemical-Kinetic Systems

PATRICK JÄMMER*

School of Chemistry, The University of Birmingham, Edgbaston, Birmingham B15 2TT, UK

Received 14 April 1997; accepted 20 June 1997

ABSTRACT: To make predictive statements about the role and reactions of different species in physicochemically reactive systems, it is necessary to formulate and solve a set of coupled differential equations describing the systems kinetic behavior. A symbolic algebra module, `eqParse.m`, is described, which parses an input file containing a set of chemical reactions, rate constants, source and sink terms, and initial conditions. From this information a ratelaw equation for each species is generated; the set of these equations may then be solved for the system's time evolution. In this article, the general mathematical framework for the treatment of such chemical-kinetic systems is first outlined. A description of the user interface is then given, with a brief discussion of the limitations inherent in such symbolic and numerical methods. The general usefulness of the proposed protocol is then demonstrated in both symbolic and numerical computations in a variety of investigations in organic reaction mechanism analysis and oscillatory combustion reactions and chaotic behavior in autocatalysis. The computed results are shown to be in good accord with experimental observations, and conclusions are drawn from these. *J Comput Chem* **18**: 1903–1917, 1997

Keywords: chemical kinetics; ratelaw; symbolic algebra; combustion; autocatalysis; reaction mechanism

*Present address: Physical and Theoretical Chemistry Laboratory, University of Oxford, South Parks Road, Oxford OX1 3QZ, UK; e-mail: padz@physchem.ox.ac.uk; URL: http://berne.pcl.ox.ac.uk/~padz/eqParse.html

Contract/grant sponsor: University of Sussex
Contract/grant sponsor: University of Birmingham
Contract/grant sponsor: EPSRC

CCC 0192-8651 / 97 / 151903-15

Introduction

A generalized reaction (r), with reactants, $\{R_i\}$, becoming products, $\{P_j\}$, where the respective stoichiometric coefficients are $\{\nu_i^r, \nu_j^r\}$, and the forward and backward rate constants are $\{k_r, k_{-r}\}$, may be written as:

$$\sum_i \nu_i^r R_i \overset{r}{\rightleftharpoons} \sum_j \nu_j^r P_j \tag{1}$$

In general, the spatial and temporal variation of the concentration, $C_n(x,t)$, of species, n, is described by a kinetic *ratelaw* of the form:

$$\frac{\partial C_n(x,t)}{\partial t} = Q_n(\{C_m\};t) + S_n(x,t) + T_n(u, D_n; x) \tag{2}$$

The various contributions include:

- **Reaction terms,** $Q_n(\{C_m\};t)$. In this framework, all the terms are thought of as modifications of a ''reaction'' rate type term. This is given by:

$$Q_n(\{C_m\};t) = \sum_r Q_n^r(\{C_m\};t) \tag{3}$$

$$Q_n^r(\{C_m\};t) = k_r \prod_p (C_p)^{\nu_p^r} \tag{4}$$

  where the sum runs over all reactions (r), producing or removing species n, and a convention is used such that production reactions have a positive sign, and destruction reactions a negative sign. It is these terms which, in general, couple in the concentrations, $\{C_m\}$, of the other reactive species, $\{m\}$.

The rate constants, $\{k_r\}$, may be constant for each species, or may vary in time; for example, a bacterial death rate which varies approximately sinusoidally with solar irradiance:

$$k_r(t) = a\sin(2\pi ft) \tag{5}$$

- **Source terms,** $S_n(x,t)$. Each species, m, may be assigned its own source term, $S_m(x,t)$. These may be constant, or space- or time-varying. An example is a spatially localized constant load with weak sinusoidal variations:

$$S(t) = a + b\sin(2\pi ft) \tag{6}$$

  In the formalism of this study, source terms are described simply as one-species reactions, p, with a ''dummy'' left-hand side, W_p:

$$W_p \overset{p}{\rightarrow} P_k \tag{7}$$

  Such terms are introduced using the keywork `source` to represent the source input rate, k_p. The ''dummy'' left-hand-side term, W_p, is factored out on forming the rate equations.

- **Transport terms,** $T_n(u, D_n; x)$. In general, transport will involve *advection* of the bulk fluid at flowrate u, and also microscopic *diffusion* of the individual particles in the fluid at a rate determined by the diffusion coefficient of each species, D_n. Thus the total transport term may be separated into two components, T_{1n} and T_{2n}, which depend on the gradient and the curvature of the flowfield, respectively:

$$T_n(u, D_n; x) = T_{1n}(u; x) + T_{2n}(D_n; x) \tag{8}$$

- **Sink terms.** It should be noted here that one sometimes wishes to consider a specific reaction, q, without regard to the products. This may be considered straightforwardly as a special case of eq. (1), in which the right-hand side is just an undefined ''dummy'' *sink* term, X_q:

$$\sum_k \nu_k^q R_k \overset{q}{\rightarrow} X_q \tag{9}$$

  The concentrations corresponding to X_q are not included in the build-up of the ratelaw equations. An example of such a system might be a partial radioactive decay chain, where only a few successive members of the series need be considered.[3] Such terms, with the sink rate constant k_q, are introduced straightforwardly using the keywork `sink` to represent the dummy right-hand side: this does not appear in the final rate equations.

Static Systems

If the transport terms are neglected, it is possible to formulate the set of corresponding ordinary differential equations for the system:

$$\frac{dC_n(t)}{dt} = S_n(t) + Q_n(\{C_m\}; t) \qquad (10)$$

For simple systems containing only few reactant species and competing kinetic processes, the ratelaws may be set up fairly straightforwardly, and are amenable to analytic solution. However, for more complicated examples, there may be many species in the system, which interact in complex ways. This results in a system of many coupled differential equations too difficult to handle manually, even after neglecting transport effects. To address this problem, a *Mathematica*[4–9] symbolic algebra module, `eqParse.m`, was devised—because this approach has been shown to be highly useful in a variety of previously studied applications of this kind.[10–13] This parses an input file containing the raw ''chemical equations,'' in the form of eq. (1); it then forms the appropriate governing ratelaw equations [eq. (10)] and solves these for the two-evolution of each species.

Overall, the system is treated as a *black box* with its input and output being the species of interest. No attempt is made to describe the detailed microscopic processes involved in chemical reactivity. The most influential reactions have large rate constants and involve species present in high concentrations.

The remainder of this article gives examples of the usefulness of `eqParse.m` in a variety of chemical kinetic contexts. The computed solutions for the systems studied are compared in each case with the corresponding experimental observations.

User Interface

To use the novel *Mathematica* module, `eqParse.m`, one needs to obtain the source code, listed in Appendix 5, which is available via the World-Wide Web.[14] This module acts as an ''add-on,'' and is designed to exploit the suite of mathematical functionality already available in *Mathematica*.[9] It combines the methodology outlined in previous work[10–13] into a coherent package, to aid the analysis of chemical kinetic systems.

The raw ''chemical equations,'' together with information on the boundary conditions and number of species and reactions, is given by the user in a file called `infile`. The template for setting up `infile` is given in Appendix 0; the input files used to generate the output for each of the examples are also given in the appendices.

Consider the simple example of a three-species, two-step reaction:

$$A \xrightarrow{1} B \xrightarrow{2} C \qquad (11)$$

Each of the species, A and B, decays into the next in a first-order manner. In the above scheme, reaction i has forward rate constant k_i, which is assumed to be constant in time. For simplicity, the initial concentrations are taken to be `A(0)` = A_0, `B(0)` = `C(0)` = `0`.

The user initially creates `infile`. The first block of four items entered consists of the number of equations, `neq`, the number of species, `nsp`, the species names in the form `A[t]` for species, A, and the initial conditions, in the form `A[0]` == `A0`. For this example, one has:

```
2
3
A[t] B[t] C[t]
A[0] == A0 B[0] == 0 C[0] == 0
```

The next block of input consists of the source, sink, and reaction terms for each step in the mechanism, including the rate constants. In the above example, for which $A \rightarrow B$ and then $B \rightarrow C$, and in which there are no source or sink terms, one obtains:

```
A[t]
B[t]
k1
B[t]
C[t]
k2
```

The *Mathematica* kernel is started, and the user then loads the module by typing, at the prompt:

```
<<eqParse.m
```

before the information in `infile` is parsed by giving the command:

```
eqParse[infile];
```

thus creating a list of equations and initial conditions, `eq`, and a list of reacting species, `sp`.

To effect a solution, use is made of the built-in *Mathematica* routines for solving a set of ordinary differential equations, either purely symbolically with `DSolve`, or numerically using `NDSolve`.

It is very important to remember at this stage that only relatively small equation systems may be solved symbolically, due to the intrinsic limitations in inverting symbolic matrices of dimension greater than 4×4. So, for *completely general* symbolic systems, the user is restricted to four species and four equations.[9] In special cases, however, simplifications are possible that allow larger systems to be dealt with symbolically. To attempt an algebraic solution, the user types:

```
eqSolve[]
```

and, if the system is soluble, output is returned in the form:

```
{{A[t]->a1, B[t]->b1, C[t]->c1}}
```

where `a1`, `b1`, and `c1` are the symbolic solutions. If, however, the system of equations is not soluble symbolically, the kernel simply returns the `DSolve` call with `eq` and `sp` substituted with values derived from `infile`.

However, there is much greater scope for numerical solution of the general problem. Again, however, there is no choice of the specific methodology used in generating such a solution; this is chosen by the *Mathematica* kernel in the light of each problem. All examples given below are amenable to solution by the methods built into *Mathematica*. Nevertheless, there are still some systems that require special solution techniques (e.g., stiff equation systems arising in enzyme kinetics[15–17]). To get a numerical solution, if possible, over the time range `tmin` to `tmax`, the user types:

```
eqNSolve[tmin, tmax]
```

and in this case, if soluble, a solution is returned in the form:

```
{{A[t]->InterpolatingFunction[{{tmin,
   tmax}},<>][t],
  B[t]->InterpolatingFunction[{{tmin,
   tmax}},<>][t],
  C[t]->InterpolatingFunction[{{tmin,
   tmax}},<>][t]}}
```

If the problem is not soluble as posed then the `NDSolve` call is returned with `eq` and `sp` replaced with the values as parsed from `infile`.

After a solution has been obtained it is possible to use all other built-in features of *Mathematica* to perform postprocessing of the solution; this might include further symbolic manipulation, for example, or graphing the time evolution of the species. The overall set of instructions to parse the input, obtain a solution, and further process the result, in each of the examples considered next, is given in a separate listing as a *runfile* in appendices 1, 2, 3 and 4.

EXAMPLE 1: COMPLEX DEHYDRATION MECHANISM

The complex dehydration mechanism is an example of the general mechanism eq. (11), specifically that of prostaglandin E_1 methyl ester dehydration, as discussed by Connors.[18] The input file, `input`, for this problem is given in Appendix 1.1, and the *Mathematica* runfile in Appendix 1.2.

If one defines the quantities $\Delta_{ij} = k_i - k_j$, $f_{ij} = \exp(k_i t) - \exp(k_j t)$ and $g_i = 1 - \exp(-k_i t)$, then the solution generated may be written:

$$A(t) = A_0 \exp(-k_1 t) \tag{12}$$

$$B(t) = \frac{A_0 k_1 f_{21}}{\Delta_{12}} \tag{13}$$

$$C(t) = \frac{A_0}{\Delta_{12}} [k_1 g_2 - k_2 g_1] \tag{14}$$

Thus, it can be shown that, for such a reaction, a hierarchy of solutions exists, which for the above case is expressible in a closed form. For this particular example it is actually straightforward to generalize this methodology to more complex cases and obtain analytic solutions.

In terms of prostaglandin E_1 methyl ester dehydration, the above symbolic solution may be used to investigate a specific experimental case, the reaction at pH 7.66 and temperature 60°C. In this case, the rate constants would be $k_1 = 0.087\ h^{-1}$ and $k_2 = 0.0020\ h^{-1}$. To derive concentrations relative to the initial concentration, the value $A_0 = 1$ was chosen. The time evolution of the species concentrations is shown in Figure 1, and this can be seen to be in very good agreement with the experimental observations given by Connors [18].

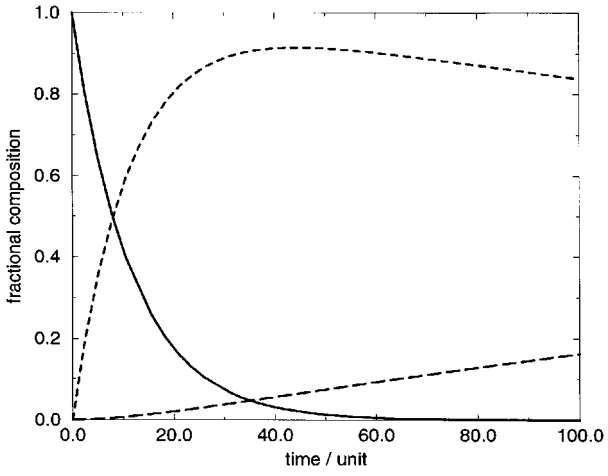

FIGURE 1. Prostaglandin E_1 methyl ester dehydration. pH 7.66, $T = 60°C$, $k_1 = 0.087\ h^{-1}$, $k_2 = 0.0020\ h^{-1}$, $A_0 = 1$. The solid line is for `A[t]`; the short-dashed line is for `B[t]`; the long-dashed line is for `C[t]`.

EXAMPLE 2: COMPLEX DEHALOGENATION MECHANISM

Consider next the slightly more complex example of a four-species, three-step reaction:

$$AX \xrightarrow{1} BX \xrightarrow{2} C \xrightarrow{3} F \tag{15}$$

Each of the species, AX, BX, and C decays into the next in a first-order manner. In the above scheme, reaction i has the forward rate constant k_i, which is assumed to be constant in time. For simplicity, the initial concentrations are taken to be $AX(0) = A_0$, $BX(0) = C(0) = F(0) = 0$. An example of this mechanism is given by Liu et al. [19], where the investigators discuss the relative rates of dehalogenation of a complex organic molecule, AX, in which the halogen atom, X, may be Cl, Br, or I. The input file, `input`, for this problem is given in Appendix 2.1, and the *Mathematica* runfile in Appendix 2.2.

If one uses the quantities, Δ_{ij}, f_{ij}, and g_i, defined previously, the solutions may be written:

$$AX(t) = A_0 \exp(-k_1 t) \tag{16}$$

$$BX(t) = \frac{A_0 k_1 f_{21}}{\Delta_{12}} \tag{17}$$

$$C(t) = \frac{A_0 k_1 k_2}{\Delta_{12}\Delta_{13}\Delta_{23}}[k_1 f_{32} + k_2 f_{13} + k_3 f_{21}] \tag{18}$$

$$F(t) = \frac{A_0}{\Delta_{12}\Delta_{13}\Delta_{23}}\left[k_1 k_2 g_1 \Delta_{23} + k_1^2(k_2 g_3 - k_3 g_2) + k_1(k_3^2 g_2 - k_2^2 g_3)\right] \tag{19}$$

This solution can be seen to be a generalization of the example considered in the previous section.

In terms of the study [19], at pH 7.5 and temperature 25°C, numerical results were obtained using the experimental rate constants for X = Cl, Br, and I. For each case, the final product species' time evolution was calculated, and the three curves are shown in Figure 2. These calculations verify the investigators' assertion that the rate of product formation is determined by the ease of formation of the intermediate species, C, which is easiest when the leaving group, X, is I, and most difficult when it is Cl.

EXAMPLE 3: SOLID-PHASE COMBUSTION

The following mechanism has been proposed to represent the situation in solid-phase pyrotechnic combustion of magnesium [20], and leads to interesting oscillatory behavior:

$$A \xrightarrow{3} B + F \quad A + 2B \xrightarrow{1} 3B \quad B \xrightarrow{2} F \tag{20}$$

In the above scheme, reaction, i, has a forward rate constant, k_i, which is assumed to be constant in

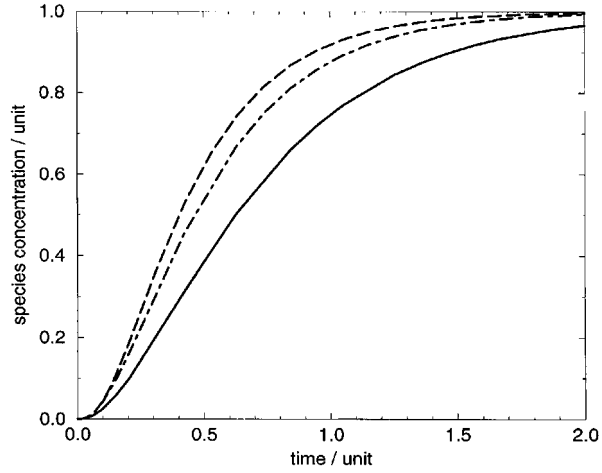

FIGURE 2. 4-Halobenzoyl coenzyme A dehalogenation. Time variation of `F[t]`. The solid line is for the case X = Cl; the dot-dashed line is for the case X = Br; the long-dashed line is for the case X = I.

time. In the physical example, A represents O_2, B represents Mg (g), and F represents MgO.

The input file, `infile`, containing the physical parameters of ref. 20, is given in Appendix 3.1, and the *Mathematica* runfile in Appendix 3.2.

The time evolution of the concentrations of species A and B is given in Figure 3. As observed experimentally, the concentrations of these species cycle synchronously, with some damping over the time scale studied. This behavior is taken as being an integral part of the system's pyrotechnic reactivity.

EXAMPLE 4: HETEROGENEOUS AUTOCATALYSIS

This section discusses another example of a mechanism proposed to represent many complex physical situations, such as the oxidation reactions on a metal surface of CO, H_2, CH_3CHO, or C_mH_n.[21] Such systems are particularly interesting because they can exhibit chaotic behavior under stable physical conditions:[21]

$$P \xrightarrow{0} A \quad P + C \xrightarrow{0} A + C \quad A \xrightarrow{v} B$$
$$A + 2B \xrightarrow{1} 3B \quad B \xrightarrow{2} C \quad C \xrightarrow{3} F \qquad (21)$$

In the above scheme, reaction i has the forward rate constant k_i, which is assumed to be constant in time. Rate constants corresponding to such physical situations are quoted in ref. 21, and these are used here to investigate the behavior of such systems. The input file, `infile`, containing the physical parameters of ref. 21, is given in Appendix 4.1, and the *Mathematica* runfile in Appendix 4.2.

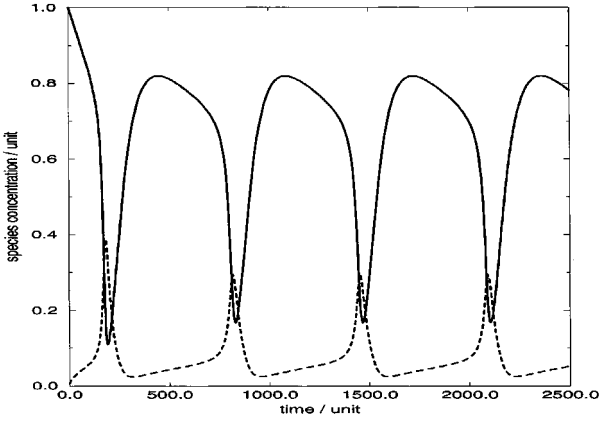

FIGURE 3. Solid-phase combustion. The solid line is for `A[t]`; the short-dashed line is for `B[t]`.

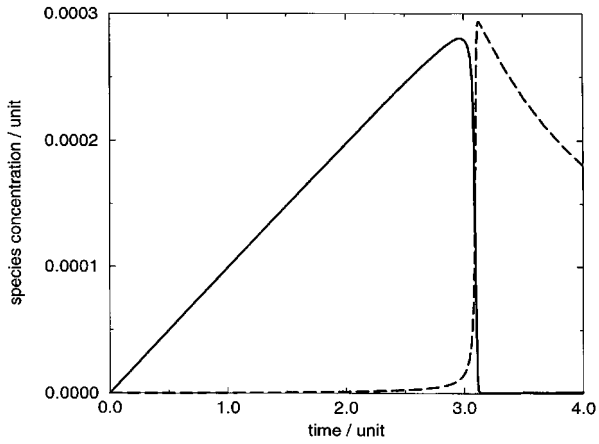

FIGURE 4. Heterogeneous autocatalysis. The solid line is for `A[t]`; the long-dashed line is for `B[t]`.

The catastrophic behavior exemplified by these systems, as discussed and rigorously characterized in ref. 21, is clearly seen in the sharp step change in the concentration variation at $t = 3$, as plotted in Figures 4 and 5.

Summary

A *Mathematica* module, `eqParse.m`, has been presented in the context of chemically reactive systems analysis. This is designed to calculate symbolically, or for specific numerical scenarios, the time variation of the species concentrations. Input is given in the familiar ''chemical equation'' format for the reactions and their rate constants, together with the initial species concentrations. The

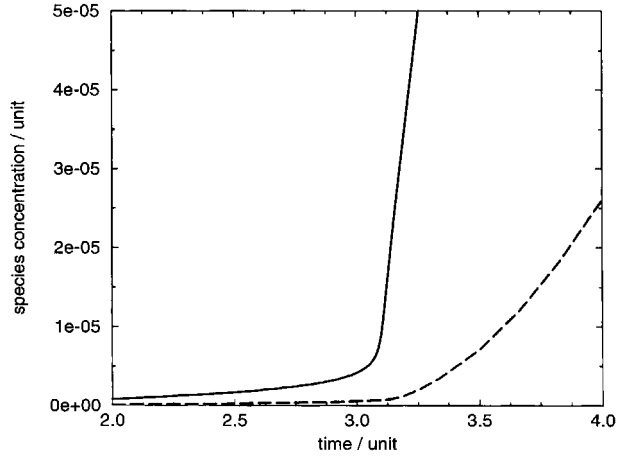

FIGURE 5. Heterogeneous autocatalysis. The long-dashed line is for `C[t]`; the long-dashed line is for `F[t]`.

usefulness of the method lies in its ability to provide an intuitively simple and practically useful tool for computations that are difficult and tedious to do by hand. Moreover, the extra functionality built into *Mathematica* allows for specific symbolic processing of the resulting solutions.

Examples have been presented to show how this method can be used extensively in the investigations of ostensibly different chemical systems.

Work in preparation for publication will detail the extension of this approach to the reaction-transport case, using a *plug flow* regime for advection and diffusion, and it will apply this to bacterial population dynamics.[22]

Acknowledgment

The author thanks the referees for their helpful comments and suggestions in refining the work presented here.

Appendix 0: Input File Template

- The format of the input file, `infile`, follows the pattern shown below.
- The input is made up of four initial statements consisting of the number of equations, the number of species, the species names as functions of time, and the initial species concentrations.
- This is followed by a *carriage return* or *newline*, represented by `<cr>` in the examples below.
- Then, follows text in three-line *blocks*.
- Each *block* is separated by a *newline*.
- A convention is used whereby any *block* of three consecutive lines enclosed in braces is optional and, if used, can be repeated as often as desired.
- Other single text items in braces are optional.
- A straightforward notation for the species and other quantities is used. For example, in coding the input for the subsection, ''Example 1: Complex Dehydration Mechanism,'' the following correspondences are used: `A0` and `k1` represent A_0 and k_1, respectively; and `B[t]` is the time-varying concentration, $B(t)$.

```
neq                          no. equations (integer >= 1)
nsp                          no. species (integer >= 1)
S1[t] {S2[t]}                reacting species
S1[0] == S10 {S2[0] == S20}  initial conditions
<cr>
{ source     }               source term label
{ Sa[t]      }               species created by source
{ k_source   }               source input rate
{<cr>        }
  Sb[t] {Sc[t]}              reactant species
  Sd[t] {Se[t]}              product species
  k1                         rate constant
<cr>
{ Sb[t] {Sc[t]}
{ Sd[t] {Se[t]}
{ k1         }
{<cr>
{ Sz[t]      }               species removed by sink
{ sink       }               sink term label
{ k_sink     }               sink removal rate
```

Appendix 1

APPENDIX 1.1: INPUT FOR SUBSECTION "EXAMPLE 1: COMPLEX DEHYDRATION MECHANISM"

```
2
3
A[t] B[t] C[t]
A[0] == A0 B[0] == 0 C[0] == 0
A[t]

B[t]
k1

B[t]
C[t]
k2
```

APPENDIX 1.2: RUNFILE FOR SUBSECTION "EXAMPLE 1: COMPLEX DEHYDRATION MECHANISM"

```
(* load the module *)
<<eqParse.m;
(* parse the input *)
eqParse[infile];
(* symbolically solve for concentrations *)
ans = eqSolve[] // Simplify;
(* symbolic simplification rules *)
sub4 := ((-1 + Exp[zz_*t])*Exp[-zz_*t])->(1-Exp[-zz*t]);
sub3 := (yy_-yy_/Exp[zz_*t])->yy*(1-Exp[-zz*t]);
sub2 := {(Exp[-aa_*t]-Exp[-bb_*t])->f[aa,bb]};
sub1b := (1-Exp[-dd_*t])->-gg[dd];
sub1a := (-1 + Exp[-dd_*t])->g[dd];
sub0 := (ee_-ff_)->delta[ee,ff];
smp4 = ans //.sub4;
smp3 = smp4 //.sub3;
smp2 = smp3 //.sub2;
smp1 = smp2 //.sub1b;
smp0 = smp1 //.sub1a;
(* tabulate results *)
nmr = Numerator[sp/.smp0][[1]];
dnm = Denominator[sp/.smp0][[1]];
Table[nmr[[dfg]] /(dnm[[dfg]]//.sub0),
{ dfg,1,3]}//TableForm
(* experimental case: pH = 7.66; T = 60°C *)
sbs:=
{
k1->0.087,
k2->0.0020,
A0->1.0
 };
sln = ans/.sbs//Flatten;
cnc = sp/.sln;
Plot[sp,{t,0,100}]
```

Appendix 2

APPENDIX 2.1: INPUT FOR SUBSECTION "EXAMPLE 2: COMPLEX DEHALOGENATION MECHANISM"

```
3
4
AX[t] BX[t] C[t] F[t]
AX[0] == A0 BX[0] == 0 C[0] == 0 F[0] == 0
AX[t]
BX[t]
k1
BX[t]
C[t]
k2
C[t]
F[t]
k3
```

APPENDIX 2.2: RUNFILE FOR SUBSECTION "EXAMPLE 2: COMPLEX DEHALOGENATION MECHANISM"

```
(* load the module *)
<<eqParse.m;
(* parse the input *)
eqParse[infile];
(* symbolically solve for concentrations *)
ans = eqSolve[] // Simplify;
(* symbolic simplification rules *)
sub4 := ((-1 + Exp[zz_*t])*Exp[-zz_*t])->(1-Exp[-zz*t]);
sub3 := (yy_-yy_/Exp[zz_*t])->yy*(1-Exp[-zz*t]);
sub2 := {(Exp[-aa_*t]-Exp[-bb_*t])->f[aa,bb]};
sub1b := (1-Exp[-dd_*t])->-gg[dd];
sub1a := (-1 + Exp[-dd_*t])->g[dd];
sub0 := (ee_-ff_)->delta[ee,ff];
smp4 = ans //.sub4;
smp3 = smp4 //.sub3;
smp2 = smp3 //.sub2;
smp1 = smp2 //.sub1b;
smp0 = smp1 //.sub1a;
(* tabulate results *)
nmr = Numerator[sp/.smp0][[1]];
dnm = Denominator[sp/.smp0][[1]];
Table[nmr[[dfg]]/(dnm[[dfg]]//.sub0),
{dfg,1,4}]//TableForm
(* experimental study: pH = 7.5; T = 25°C *)
(* reaction 1: X = Cl *)
sbscl:=
{A0->1, k1->67, k2->3.8, k3->2.1};
slncl = ans/.sbscl//Flatten;
cnccl = sp/.slncl;
```

```
ccl = cnccl[[4]];
(* reaction 2: X = Br*)
sbsbr :=
{A0->1, k1->58, k2->5.2, k3->2.9};
slnbr = ans /. sbsbr // Flatten;
cncbr = sp /. slnbr;
cbr = cncbr[[4]];
(*reaction 3: X = I*)
sbsid :=
{A0->1, k1->25, k2->7.5, k3->3.2};
slnid = ans /. sbsid // Flatten;
cncid = sp /. slnid;
cid = cncid[[4]];
(* plot concentrations *)
Plot[{ccl,cbr,cid},{t,0,5}];
```

Appendix 3

APPENDIX 3.1: INPUT FOR SUBSECTION "SOLID-PHASE COMBUSTION"

```
5
3
A[t] B[t] F[t]
A[0] == 1 B[0] == 0.006 F[0] == 0
A[t]
B[t] F[t]
1 / 650
A[t] B[t] B[t]
B[t] B[t] B[t]
1 / 1 ^2
B[t]
F[t]
1 / 20
source
A[t]
(1.0 - A[t]) / 74
source
B[t]
(0.006 - B[t]) / 74
```

APPENDIX 3.2: RUNFILE FOR SUBSECTION "SOLID-PHASE COMBUSTION"

```
(* load the module *)
<<eqParse.m;
(* parse the input *)
eqParse[infile];
(* numerically solve for concentrations *)
ans = eqNSolve[0,2500] // Simplify;
```

```
sln = sp /. ans // Flatten;
aa = sln[[1]];
bb = sln[[2]];
(* plot concentration variation *)
Plot[{aa,bb},{t,0,2500}];
```

Appendix 4

APPENDIX 4.1: INPUT FOR SUBSECTION "HETEROGENEOUS AUTOCATALYSIS"

```
6
5
P[t] A[t] B[t] C[t] F[t]
P[0] == 0.1 A[0] == 0 B[0] == 0 C[0] == 0 F[0] == 0
P[t]
A[t]
0.001
P[t] C[t]
A[t] C[t]
0.001
A[t]
B[t]
0.01
A[t] B[t] B[t]
B[t] B[t] B[t]
2.5*10^9
B[t]
C[t]
1
C[t]
F[t]
0.25
```

APPENDIX 4.2: RUNFILE FOR SUBSECTION "HETEROGENEOUS AUTOCATALYSIS"

```
(* load the module *)
<<eqParse.m;
(* Parse the input *)
eqParse[infile];
(* numerically solve for concentrations *)
ans = eqNSolve[0,10] // Simplify;
sln = sp /. ans // Flatten;
aa = sln[[2]];
bb = sln[[3]];
cc = sln[[4]];
ff = sln[[5]];
(* plot concentration variation *)
Plot[{aa,bb},{t,0,4}];
Plot[{cc,ff},{t,0,4}];
```

Appendix 5: Module eqParse.m

```
(* Copyright 1997 P. Jemmer *)
(* $Revision 3.0$ *)
BeginPackage["eqParse`"]
eqParse::usage=
"eqParse[infile] parses the chemical kinetics
equations in infile into ODE form."
eqSolve::usage=
"eqSolve[]
analytically solves the generated ODEs in variable var."
eqNSolve::usage=
"eqSolve[varmin,varmax]
numerically solves the generated ODEs in variable var
over the range varmin to varmax."
(* module user information *)
(* parsing instructions *)
messagestring:=
"n
#####################################\n
#                                    #\n
##Module:                 _eqParse.m_##\n
##Inputfile:               _infile_  ##\n
#                                    #\n
##This module parses an inputfile    ##\n
##containing chemical equations      ##\n
#                                    #\n
##To parse the required input file##\n
##use the command                    ##\n
#                                    #\n
###In[2]:= eqParse[infile]          ###\n
#                                    #\n
#####################################\n
";
(* print module user information *)
Print[messagestring];
(* define parsing function *)
eqParse[inputfile_]:=
(
Module[
{
fnzz,ipzz,eqzz,spzz,bczz,dfzz,
sozz,sizz,
i1zz,j1zz,t1zz,s1zz,v1zz,w1zz,
i2zz,j2zz,t2zz,s2zz,v2zz,w2zz,
azz,bzz,czz,dzz,ezz,fzz,gzz,hzz,
izz,jzz,kzz,lzz,mzz,nzz,pzz,qzz,
rzz,szz,tzz,uzz,vzz,wzz,xzz,yzz,
endstringzz
 },
{
```

```
fnzz = ToString[inputfile];
ipzz = OpenRead[fnzz];
eqzz = Read[ipzz,Number];
spzz = Read[ipzz,Number];
sp = ToExpression[ReadList[ipzz,Word,spzz]];
bczz = ToExpression[ReadList[ipzz,Word,spzz]];
(* initialize lists *)
Do[
{
fzz[vzz] = Array[0,3],
sozz[vzz] = 0,
sizz[vzz] = 0
 },
{vzz,1,eqzz}
 ];
gzz = ReadList[ipzz,Word,RecordLists->True];
nzz = 0;
Do[
{
Do[
{
azz = ToExpression[Part[gzz,vzz+ 2*nzz + czz- 1]]//Flatten,
fzz[vzz] = ReplacePart[fzz[vzz],azz,czz]
 },
{czz,1,3}
 ],
nzz += 1
 },
{vzz,1,eqzz}
 ];
Close[ipzz];
dzz = Table[0,{bzz,1,spzz}];
Do[
{
lzz[wzz] = Length[Part[fzz[wzz],1]],
mzz[wzz] = Length[Part[fzz[wzz],2]],
kzz[wzz] = Part[Part[fzz[wzz],3],1],
 },
{wzz,1,eqzz}
 ];
Do[
{
If[
fzz[jzz][[1]] == List[source],
kzz[jzz] = Part[Part[fzz[jzz],3],1]/source,
kzz[jzz] = Part[Part[fzz[jzz],3],1]
 ],
If[
fzz[jzz][[1]] == List[source],
sozz[jzz] = kzz[jzz]*source,
sozz[jzz] = 0
 ],
If[
fzz[jzz][[2]] == List[sink],
```

```
sizz[jzz]=-kzz[jzz],
sizz[jzz]=0
 ];
 },
{jzz,1,eqzz}
 ];
so=Table[sozz[yzz],{yzz,1,eqzz}];
si=Table[sizz[yzz],{yzz,1,eqzz}];
Do[
{szz=lzz[vzz],
rzz=mzz[vzz],
xzz=Product[Part[Part[fzz[vzz],1],izz],{izz,1,sz}],
qzz=kzz[vzz]*xzz,
Do[
Do[
{
t1zz=Part[Part[fzz[vzz],1],j1zz],
s1zz=sp[[i1zz]],
v1zz=dzz[[i1zz]],
If[t1zz===s1zz,w1zz=v1zz-qzz,w1zz=v1zz],
dzz[[i1zz]]=w1zz
 },
{i1zz,1,spzz}
 ],
{j1zz,1,szz}
 ],
Do[
Do[
{
t2zz=Part[Part[fzz[vzz],2],j2zz],
s2zz=sp[[i2zz]],
v2zz=dzz[[i2zz]],
If[t2zz===s2zz,w2zz=v2zz+qzz,w2zz=v2zz],
dzz[[i2zz]]=w2zz
 },
{i2zz,1,spzz}
 ],
{j2zz,1,rzz}
 ]
 },
{vzz,1,eqzz}
 ];
var=sp[[1]][[1]];
dfzz=Table[Dt[sp[[uzz]],var],{uzz,1,spzz}];
tzz=Table[dzz]//Flatten;
hzz=Table[dfzz[[ezz]]==tzz[[ezz]],{ezz,1,spzz}];
eq=AppendTo[hzz,bczz]//Flatten;
(* how to solve {ep,bc} set for sp *)
(* print user instructions *)
endstringzz:=
"
########################################\n
#                                      \n
## The equation parsing is completed  ##\n
```

```
#                                          #\n
## To solve for reacting species          ##\n
## use the following syntax               ##\n
#                                          #\n
## (1) for analytic solution              ##\n
#                                          #\n
### In[3]:= eqSolve[]                     ###\n
#                                          #\n
#                                          #\n
## (2) for numerical solution over        ##\n
##  range (varmin, varmax)                ##\n
#                                          #\n
### In[3]:= eqNSolve[varmin,varmax]      ###\n
#                                          #\n
##########################################\n
";
Print[endstringzz];
eqSolve[]:= DSolve[eq,sp,var];
eqNSolve[varmin_,varmax_]:= NDSolve[eq,sp,{var,varmin,varmax}]
 }
 ]
 )
EndPackage[]
Null
```

References

1. I. Amdur, *Chemical Kinetics*: *Principles and Selected Topics*, McGraw-Hill, New York, 1966.
2. M. Boudart, *Kinetics of Chemical Processes*, Butterworth-Heinemann, Boston, MA, 1991.
3. N. N. Greenwood and A. Earnshaw, *Chemistry of the Elements*, Pergamon, Oxford, 1984.
4. A. I. Beltzer, *Engineering Analysis with Maple/Mathematica*, Academic Press, New York, 1995.
5. N. Blachman, *Mathematica, a Practical Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1992.
6. R. E. Crandall, *Mathematica for the Sciences*, Addison-Wesley, Reading, MA, 1991.
7. W. Ellis, *A tutorial introduction to Mathematica*, Brooks/Cole, Monterey, CA, 1991.
8. R. E. Maeder, *The Mathematica Programmer*, AP Professional, San Diego, CA, 1994.
9. S. Wolfram, *Mathematica*: *A System for Doing Mathematics by Computer*, Addison-Wesley, Reading, MA, 1992.
10. M. Farza and A. Cheruy, *Comput. Appl. Biosc.*, **10,** 477 (1994).
11. C. D. Stoner, *Biochem. J.*, **291** 585 (1993).
12. E. Urbanovici, H. A. Schneider, D. Brizzolara, and H. J. Cantow, *J. Therm. Anal.*, **47,** 931 (1996).
13. A. R. Waldeck and R. Stocker, *Chem. Res. Toxicol.*, **9,** 954 (1996).
14. http://berne.pcl.ox.ac.uk/~padz/eqParse.html
15. C. J. Aro, *Comput. Phys. Commun.*, **97,** 304 (1996).
16. D. Schwalbe, H. Kooljman, and R. Taylor, *Mapletech*, **3,** 47 (1996).
17. P. Sun, D. P. Chock, and S. L. Winkler, *J. Comput. Phys.*, **115,** 515 (1994).
18. K. A. Connors, *Chemical Kinetics*: *The Study of Reaction Rates in Solution*, VCH, New York, 1990.
19. R.-Q. Liu, P.-H. Liang, J. Scholten, and D. Dunaway-Mariano, *J. Am. Chem. Soc.*, **117,** 5003 (1995).
20. C.-G. Feng, Q.-X. Zeng, L.-Q. Wang, and X. Fang, *J. Chem. Soc. Faraday Trans.*, **92,** 2971 (1996).
21. A. N. Chaudry, P. V. Coveney, and J. Billingham, *J. Chem. Phys.*, **100,** 1921 (1994).
22. P. J̈emmer, K. Kratz, and N. Read, submitted to *Mathematical Biosciences*.